H.264/AVC에 적용 가능한 압축영상용 CAVLC 인코더 설계

정덕영*, 최덕영*, 손승일*

한신대학교 정보통신학과

Design of CAVLC Encoder for the Compressed Image in H.264/AVC

Duck Young Jung, Dug Young Choi, Seung Il Sonh

Dept. of Information and Communication, HanShin University

E-mail : mykor01@naver.com

요 약

요즘 시대는 영상 기술과 IT 발전으로 다양한 멀티미디어 서비스를 제공하기 위해 고품질의 비디오와 높은 데이터 압축을 요구하게 되었고, 이를 위해 MPEG-4 AVC/H.264에서는 기존의 MPEG-4에서 채택한 VLC 기술과 유사한 Context-based Adaptive Variable Length Code (CAVLC) 기술을 채택하여 이를 가능하게 하였다. 특히 CAVLC 기술은 HDTV처럼 큰 영상 뿐 아니라 카메라폰이나 DMB등과 같은 영상에서 고품질의 영상을 보다 효율적으로 제공 한다.

본 논문은 최근의 이미지와 비디오 압축에 대한 요구에 따라 H.264/AVC와 MPEG4-PART 10에서 표준으로 채택한 CAVLC의 부호화 과정을 연구하여 Visual C++언어를 이용한 최적화된 시뮬레이션과 CAVLC의 성능평가를 통한 최적화를 실시하였고, 최적화된 예측 정보를 바탕으로 CAVLC를 VHDL언어를 이용하여 하드웨어로 구현하였다.

키워드

CAVLC, coeff_token, level_prefix, suffixlength

Ⅰ. 서론

영상 매체와 기술에 대한 개발로 디지털 방송과 디지털 컨텐츠들이 발전하게 되었다. 이런 발전은 비디오 압축의 표준화에 의해 가능하게 되었고, 멀티미디어 시스템에서 비디오 처리를 가능하게 하는 중요한 역할을 한다. Variable Length Code는 이미지와 비디오 압축 응용에 중요한 역할을 하며 이미지와 비디오 압축에 대한 많은 국제 규격의 필수 성분이다. 최근 Context-based Adaptive Variable Length Code (CAVLC)는 ITU의 H.264와 MPEG-4 Part 10의 AVC에서 항상 불려지는 새로운 국제 비디오 규격인 JVT에서 entropy coding method처럼 채택되었고, zigzag 4x4(2x2) 정렬의 변환 계수 블록의 나머지를 부호화하는데 사용되며, 4개의 테이블 중에서 한 개의 테이블을 선택해서 코딩하므로 테이블들에 의한 메모리가 필요하고, CAVLC의 부호화 과정에서 생기는 지연이 발생하게 된다[1][2].

본 논문에서는 CAVLC에서 발생하는 메모리 접근 알고리즘을 최소화한 결과와 level에서 사용되는 7개의 table을 참조하지 않고 수행하는 알고리즘을Visual C++언어를 사용하여 성능평가를 하고, 최적화된 결과와 pingpong RAM을 사용하여 부호화시에 발생하는 지연을 최소화한 결과를 적용하여 VHDL언어로 설계하였다.

Ⅱ. 본론

2.1 CAVLC의 개요

CAVLC는 고품질의 이미지와 비디오 압축을 하기 위한 것으로서, zigzag 스캔된 4X4(2X2) 변환 계수들의 오차 블록을 인코딩 하는데 사용되고, 양자화된 4X4블럭의 다음과 같은 특징을 이용하기 위해 만들어졌다.

첫 번째는 예측, 변환 그리고 양자화 이후에 블록들은 일반적으로 '0'을 포함하기 때문에 CAVLC는 run_beforelevel 코딩을 수행한다. 두 번째는 zigzag 스캔 이후에 '0'이 아닌 가장 큰 계수들은 대개 ±1이며, 이를 CAVLC는 Trailing Ones를 이용하여 부호화 한다. 세 번째는 인접한 블록의 '0'이 아닌 계수들의 개수는 상관관계가 있고, 이것을 이용하여 VLC look-up table을 선택하고 부호화 한다. 마지막으로 '0'이 아닌 계수들의 레벨은 재배치된 배열의 시작점에서 보다 큰 경향이 있으므로 CAVLC는 최근에 부호화된 레벨에 따라 부호화 한다[3][4].

2.2 CAVLC 인코딩 과정

4X4블럭에서 계수의 갯수와 trailing ones를 인코딩을 하고 trailingOne의 부호를 인코딩한 후

나머지 '0'이 아닌 계수들의 레벨을 인코딩한다. 그리고 마지막 계수 이전의 전체 '0'의 갯수를 인코딩하고, 그 결과를 이용하여 각 '0'의 run을 인코딩한다.

2.3 CAVLC 구문 요소

2.3.1. coeff_token(total_coeff)

0이 아닌 전체 계수(TotalCoeffs)와 ±1의 개수(TrailingOnes)를 4x4 블록에 대하여 부호화 한다. total_coeff는 0에서 16사이의 값을 가지게 되고, T1은 0에서 최대 3까지 값을 가지며 3이상일 경우 오직 3개만 특별한 경우로 취급되고 나머지는 레벨에서 일반 계수처럼 부호화 된다.

coef_token을 부호화 하는데 아래 표1에서 보여주고 있는 4개의 look-up 테이블이 사용되며, 그 중에서 3개는 가변길이 코드 테이블이고, 나머지 하나는 고정길이 코드 테이블이다. 이전에 코딩된 왼쪽(nA)과 위쪽(nB)에 있는 블록에 존재하는 0의 계수들의 개수에 의해 선택된 nC에 따라서 4개의 look-up table 중 적당한 테이블을 선택하여 부호화 한다.[5]

[표1] total_coeff table

Trailing Ones	Total Coeff	vlc0	vlc1	vlc2	vlc2
		0<=nC<2			
0	0	1	11	1111	0000 11
0	1	0001 01	0010 11	0011 11	0000 00
1	1	01	10	1110	0000 01
		. . .			
1	5	0000 0001 10	0000 110	0100 0	0100 01
2	5	0000 0010 1	0000 101	0100 1	0100 10
3	5	0000 100	0011 0	1010	0100 11
0	6	0000 0000 0111 1	0000 0011 1	0001 001	0101 00

2.3.2. sign of each T1(Trailingones)

coeff_token에서 얻은 TrailingOnes에서 가장 높은 주파수 성분의 TrailingOnes으로부터 시작하여 역순으로 +1이면 0으로 -1 이면 1 로 부호화한다.[5]

2.3.3. level coding

역순으로 부호화 되는데 가장 높은 주파수로부터 시작하여 DC 계수쪽으로 진행한다. 각 level의 부호는 접두사(level_prefix)와 접미사(level_suffix)로 구성되고, 접미사의 길이(suffixlength)는 0비트와 6비트 사이로 각각의 계속되는 코딩된 레벨의 크기에 따라 변한다. 각 level은 연속적으로 부호화된 level의 크기에 따라 7가지 VLC tabel 중에서 하나의 테이블을 선택하여 부호화 되어진다.[6][7]

2.3.4. total_zeros

재배치된 배열 내에서 0이 아닌 마지막 계수 앞의 모든 0의 합을 부호화 한다. 배열의 시작점에 많은 수의 0이 아닌 계수들을 포함하고 배열의 시작점에 있는 zero-run은 인코딩 할 필요가 없기 때문에 total zero를 나타내기 위한 별도의 VLC를 전송한다.[5]

2.3.5. run_before

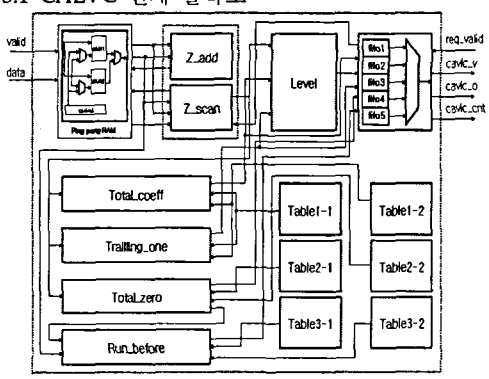
0이 아닌 계수 앞의 0의 개수를 역순으로 부호화 한다. 단, 인코딩 할 0이 더 이상 없는 경우와 0이 아닌 마지막 계수 값일 경우 예외로 한다.[5]

[표2] run_before table

run_before	zerosLeft						
	1	2	3	4	5	6	>6
0	1	1	11	11	11	11	111
1	0	01	10	10	10	000	110
2	-	00	01	01	011	001	101
3	-	-	00	001	010	011	100
				. . .			
10	-	-	-	-	-	-	0000001
11	-	-	-	-	-	-	00000001
12	-	-	-	-	-	-	000000001
13	-	-	-	-	-	-	0000000001
14	-	-	-	-	-	-	00000000001

III CAVLC 설계

3.1 CALVC 전체 블록도

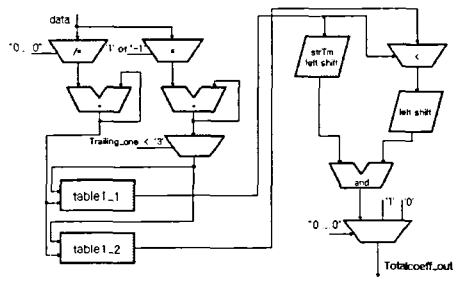


[그림 1] CAVLC 전체 블록도

그림1은 CALVC의 전체 블록도를 보여주고 있다. valid 신호가 활성화될 때 데이터를 입력받아 pingpong ram에 저장하여 부호화할 때 발생하는 지연을 최소화하고 요청신호가 활성화될 때 ram에 4×4 블록이 전송 완료되면 z_add에서 address를 전송하여 z_can에 저장하고 pingpong에서 전송된 데이터를 table1-1과 table1-2를 이용하여 total_coeff를 수행하고, 수행된 데이터를 이용하여 trailingones와 z_scan에 저장된 데이터를 이용한 level을 수행한다. 그리고 table2-1과 table2-2를 이용한 total_zero를 수행한 결과를 이용하여 run_before를 table-3과 table3-2 그리고 z_scan의 데이터를 참조하여 수행하고 각각의 모듈에서 수행된 결과를 fifo에 저장한 후 순차적으로 전송한다.

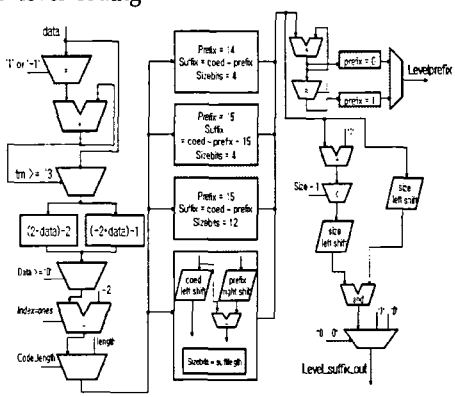
3.2 total_coeff

그림2는 total_coeff의 구성도를 보여주고 있다. 데이터를 입력받아 0이 아닌 데이터의 개수와 ±1의 개수의 결과를 이용하여 table1-1과 1-2에서 참조한 값을 이용하여 total_coeff를 부호화 한다.



[그림 2] total_coeff 구성도

3.3 level coding



[그림 3] level coding 구성도

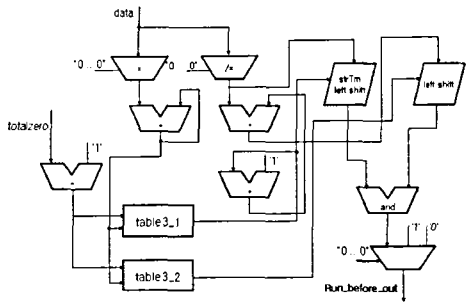
그림3은 level coding의 구성도를 보여주고 있다. 데이터를 입력받아 ±1의 개수가 3이상이면 데이터가 음수일 때와 양수일 때 levelcoed를 계산하고 계산 결과에 따라 본 논문에서 7개 table을 참조하지 않고 수행 가능한 알고리즘에 의한 levelprefix의 bit길이와 level_suffix의 bit길이를 구하고 그 값을 이용하여 level을 부호화 한다.

3.4 total_zeros

total_zeros는 데이터를 입력받아 마지막 0이 아닌 값에서 역으로 0의 개수를 구한 데이터를 이용하여 table2-1과 2-2에 참조한 값을 이용하여 total_zeros를 부호화 한다.

3.5 run_before

그림4는 run_before의 구성도를 보여주고 있다. 데이터를 입력받아 0이 아닐 때의 개수와 그렇지 않을 때 total_zero의 값을 이용하여 table3-1과 table3-2에서 참조한다. 참조한 데이터를 이용하여 알고리즘 적으로 비교하여 run_before를 부호화 한다.

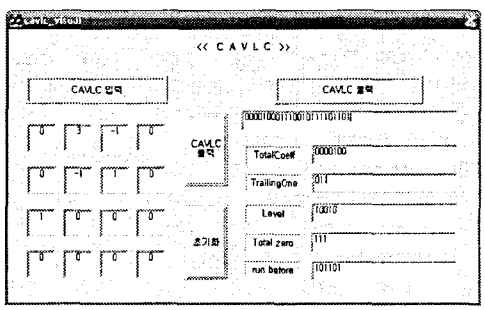


[그림 4] run_before 구성도

IV. 시뮬레이션 및 고찰

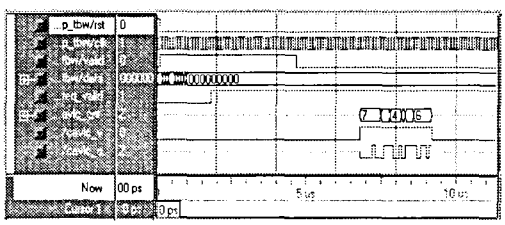
4.1 Visual C++을 이용한 시뮬레이션

아래의 그림5는 Visual C++로 시뮬레이션 하여 최적화한 결과를 보여 주고 있다. CAVLC 입력에 4×4 크기의 블록을 입력 하고 CAVLC 출력 아이콘을 클릭하면 CAVLC 출력에 전체 전송 bits가 출력되고, 각각의 구문에 따른 값을 출력한다. 출력된 각각의 구문 데이터를 쉽게 확인해 볼 수 있으며 각각의 구문에서의 에러발생시 에러정정도 용이하다.



[그림 5] Visual C++을 이용한 시뮬레이션

4.2 시뮬레이션



[그림 6] CAVLC의 인코더 출력파형

그림6은 VHDL언어를 사용하여 얻은 CAVLC 출력파형을 보여주고 있다. valid 신호가 활성화되면 4×4 블록을 입력받아 저장하고 req_valid 신호가 활성화되면 zigzag 스캔으로 재배열된 데이터를 이용하여 total_coeff, trailingones, level,

total_zeors 그리고 run_before를 3개의 table을 참조하여 구하고 처리된 데이터를 연속적으로 전송한다.

4.3 합성 결과

본 논문에서 설계한 CAVLC 인코더의 각 등가 게이트와 사용 슬라이스 수 그리고 타이밍 결과가 표4와 같이 나타났으며, total_coeff = 5, trailingones = 3일 때 24bits를 출력하는데 latency는 58 clock이다.

[표3] 설계한 CAVLC 인코더의 특징

N0. of Slices	No. of Gates	Timing	*Target* Device
674	33,633	Mininum Period: 14.544ns Maximum Frequency: 68.756MHz	*XC2V1000* -6fg256

V. 결론

영상 매체와 기술에 대한 개발로 디지털 방송과 디지털 컨텐츠들이 발전하게 되었다. 이런 발전은 비디오 압축의 표준화에 의해 가능하게 되었고, 멀티미디어 시스템에서 비디오 처리를 가능하게 하는 중요한 역할을 한다. Variable Length Code는 이미지와 비디오 압축 응용에 중요한 역할을 하며 이미지와 비디오 압축에 대한 많은 국제 규격의 필수 성분이다. H.264/AVC에서 이미지와 비디오 압축에 대해 CAVLC를 사용하는데 본 논문에서는 H.264 부호화기에서 CAVLC 부호화시에 각 구문에서 접근하는 메모리 횟수를 줄이고 level table을 참조하지 않는 알고리즘을 Visual C++언어를 이용하여 최적화된 시뮬레이션을 통한 최적화를 실시하였고, 최적화된 정보를 바탕으로 CAVLC 인코더를 VHDL언어를 이용하여 하드웨어로 구현하였다.

설계된 모듈을 이용하여 압출할 경우 처리 속도가 약간 늦다는 단점이 있지만 칩 사이즈가 줄이고 불필요한 처리를 줄여 전력소모를 줄일 수 있다. 처리 속도에 관한 연구를 좀 더 하면 H.264/AVC의 응용분야인 카메라폰의 소형화와 DMB와 같은 전력 소모가 많은 시스템에 이용이 가능하다.

참고문헌

[1] T.Wiegand, Smdy of Final Committee Draft of Joint Video Specification Draft **2,** Doc.JVT-FIOO d2, Joint Xdeo Team (IVT) of .ISO/IEC MF'EG & ITU-T VCEG Dec. 2002.

[2] GBjontcgaard and K.Lillcvold. Contest-adaptive VLC(CAVLC) coding of coefficients, Doc.JVT -028, JVT of IS0 MPEG & ITU VCEG 3' Meeting, Rairfas. Virginia, USA, May. 2002.

[3] Thomas Wiegand, Gray J. Sullivan, Gisle Bjontegaard, and Ajay Luthra, "Overview of the H.264 /AVC Video Coding Standard" *IEEE* Trans. Circuits and systems for video technology, vol.9, pp. 287-290, July. 2003.

[4] Iain E.G Richardson, "H.264 and MPEG-4", 홍릉출판사, 2004

[5] Joint Video TEam(JVT) of ISO/IEC MPEG &ITU -T VCEG (ISO/IEC JTC1/SC29/WG11 AND ITU-T SG16 Q.6) 8th Meeting: Geneva, Switzerland, 23-27 May, 2003

[6] Saied, R. Chakrabrati, C. "Scheduling for minimizing the number of memory access in low power applications" VLSI Signal Processing, IX, 1996. [Workshop on], 30 Oct.-1 Nov. 1996 Pages: 169-178

[7] 1 ITU-T Rec.H.264/ISO/iec 11496-10,"Advanced Video Coding", Final Committee Draft, Document JVT-E022, September 2002.